What is claimed is:

put AI

A processor comprising

- a first pipeline configured to execute essential code;
- a second pipeline configured to execute non-essential code; and
- 4 \quad \quad a conjugate mapping table configured to specify non-essential code to be
- 5 executed by the second pipeline.
- 1 2. The processor of claim 1 wherein the first pipeline is coupled to a first
- 2 instruction cache configured to cache instructions that determine the logical
- 3 correctness of a program.
- 1 3. The processor of claim 2 wherein the second pipeline is coupled to a second
- 2 instruction cache configured to cache instructions that provide hints for the execution
- 3 of the instructions that determine the logical correctness of the program.
- 1 4. The processor of claim 1 wherein the first pipeline is coupled to registers that
- 2 store a microarchitectural state, and wherein the conjugate mapping table is
- 3 responsive to the microarchitectural state.
- 1 5. The processor of claim 4 further comprising:
- a first instruction cache coupled to the first pipeline; and
- a second instruction cache coupled between the conjugate mapping table and
- 4 the second pipeline.
- 1 & The processor of claim 1 wherein the conjugate mapping table is responsive
- 2 to a trigger, the trigger being mapped within the conjugate mapping table to the non-

23

3 essentia code.



- 1 \(\frac{1}{\lambda}\) The processor of claim 1 wherein the conjugate mapping table comprises a
- 2 plurality of records, each of the plurality of records being configured to map a trigger
- 3 to a non-essential code sequence.
- 1 8. The processor of claim 7 wherein the trigger comprises an atomic value, such
- 2 that the conjugate mapping table is configured to specify the non-essential code
- 3 sequence when the atomic value is satisfied.
- 1 9. The processor of claim 7 wherein the trigger comprises a vector value, such
- 2 that the conjugate mapping table is configured to specify the non-essential code
- 3 sequence when the vector value is satisfied.
- 1 1d The processor of claim 1 further comprising a microarchitectural structure
- 2 counted between the first pipeline and the second pipeline.

1/2

The processor of claim 10 wherein the microarchitectural structure comprises a register bank.

- 1 /2
- A processor comprising:
- an instruction set architecture (ISA) visible path; and
- a conjugate pipeline coupled to the ISA visible path, the conjugate pipeline
- 4 being configured to execute hint calculus code and to provide execution hints to the
- 5 ISA visible path.
- 1 13. The processor of claim 12 wherein the processor can take on architectural
- 2 states and nicroarchitectural states, the conjugate pipeline being configured to affect
- 3 microarchitectural states.
- 1 14. The processor of claim 12 wherein the ISA visible path includes a pipeline
- 2 configured to execute instructions from a user program.

- 1 15 The processor of claim 14 wherein the conjugate pipeline is configured to
- 2 execute hint calculus code that virtualizes instructions from the user program.
- 1 16 \A processor comprising:
- a conjugate mapping table coupled to a first pipeline, the conjugate mapping
- table including entries to map triggers from the first pipeline to hint calculus code
- 4 sequences, and
- 5 a second pipeline to execute the hint calculus code sequences.
- 1 17. The processor of claim 16 further comprising microarchitectural structures
- 2 responsive to the second pipeline, the microarchitectural structures being configured
- 3 to modify states of the first pipeline.
- 1 18. The processor of claim 17 wherein the microarchitectural structures comprise
- 2 a branch target buffer
- 1 19. The processor of claim 17 wherein the triggers comprise:
- 2 instruction attributes;
- data attributes;
- 4 state attributes; and
- 5 event attributes.
- 1 20. The processor of claim 16 further comprising:
- an instruction cache coupled between the conjugate mapping table and the
- 3 second pipeline, the instruction cache being configured to cache the hint calculus
- 4 code sequences.

- 1 21\ The processor of claim 16 wherein the processor is configured to execute
- 2 essential code and non-essential code, and wherein the non-essential code comprises
- 3 the hint calculus code sequences.
- 1 22. The processor of claim 16 wherein the second pipeline is configured to
- 2 implement register file expansion for forward compatibility across different
- 3 generations of processors.
- 1 23. The processor of claim 16 wherein the second pipeline is configured to
- 2 implement quilt-in-self-test.
- 1 24. The processor of claim 16 wherein the second pipeline is configured to
- 2 implement instruction set virtualization.
- 1 25. A computer-implemented method of creating a runtime binary comprising:
- 2 combining an essential portion of a static binary with an essential portion of a
- 3 runtime library to speate an essential portion of the runtime binary; and
- 4 creating a non-essential portion of the runtime binary using a non-essential
- 5 portion of the static binary.
- 1 26. The computer-implemented method of claim 25 wherein creating a non-
- 2 essential portion comprises:
- 3 combining the non-essential portion of the static binary with a non-essential
- 4 portion of the runtime library.
- 1 27. The computer-implemented method of claim 26 wherein combining the non-
- 2 essential portion comprises:
- 3 combining non-essential code from the static binary with non-essential code
- 4 from the runtime library; and

5	co	nbining conjugate mapping information from the static binary with
6	conjugate	napping information from the runtime binary.

- 1 28. An article having a machine readable medium with instructions for 2 performing a method of creating a runtime binary disposed thereon, the method 3 comprising:
- combining an essential portion of a static binary with an essential portion of a runtime library to dreate an essential portion of the runtime binary; and creating a non-essential portion of the runtime binary using a non-essential portion of the static binary.
- 1 29. The article of claim 28 wherein creating a non-essential portion comprises:
 2 combining the non-essential portion of the static binary with a non-essential
 3 portion of the runtime library.
- 1 30. The article of claim 29 wherein combining the non-essential portion 2 comprises:
- combining non-essential code from the static binary with non-essential code
- 4 from the runtime library; and
- 5 combining conjugate mapping information from the static binary with
- 6 conjugate mapping information from the runtime binary.